(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2009년06월10일
(11) 등록번호 10-0901903
(24) 등록일자 2009년06월02일

(51) Int. Cl.
*G06F 11/30* (2006.01) *G06F 9/44* (2006.01)
*G06F 11/00* (2006.01)
(21) 출원번호 10-2007-7007147
(22) 출원일자 2007년03월29일
심사청구일자 2007년04월04일
번역문제출일자 2007년03월29일
(65) 공개번호 10-2007-0046963
(43) 공개일자 2007년05월03일
(86) 국제출원번호 PCT/US2005/035373
국제출원일자 2005년09월29일
(87) 국제공개번호 WO 2006/039593
국제공개일자 2006년04월13일
(30) 우선권주장
10/957,444 2004년09월30일 미국(US)
(56) 선행기술조사문헌
US20030217357 A1
US05938764 A1
US06357021 B1
전체 청구항 수 : 총 17 항

(73) 특허권자
**인텔 코오퍼레이션**
미합중국 캘리포니아 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
**카타리아, 무케쉬**
미국 95630 캘리포니아주 폴솜 블루 라빈 리지 넘버218 1005
**가프켄, 앤드류**
미국 95630 캘리포니아주 폴솜 패트릭 서클 991
**스티븐스, 윌리암, 주니어.**
미국 95630 캘리포니아주 폴솜 프리써 웨이 109
(74) 대리인
**백만기, 이중희, 주성민**

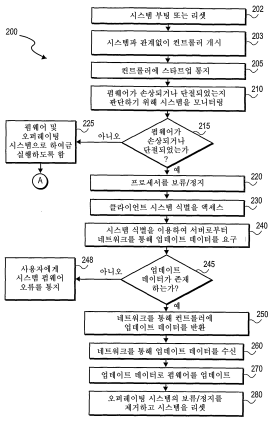
심사관 : 김견수

# (54) 네트워크를 통한 펌웨어의 셀프-모니터링 및 업데이팅

## (57) 요 약

실시예는 연산 시스템의 펌웨어가 손상, 단절되거나, 자동 업데이트를 필요로 하는지를 판단하기 위해 연산 시스템을 모니터링하는 것을 포함한다. 연산 시스템은 네트워크를 통해 펌웨어 업데이트 데이터를 요구할 수 있다. 더욱이, 연산 시스템은 펌웨어가 손상되거나 단절되었는지를 판단하고 네트워크를 통해 펌웨어 업데이트 데이터를 요구 및 수신할 수 있는 능력을 가진 컨트롤러를 포함할 수 있다. 또한, 컨트롤러는 펌웨어가 손상되거나 단절된 때, 프로세서가 정지된 경우, 오퍼레이팅 시스템이 정지, 단절, 또는 소프트-오프된 경우일지라도 작동할 수 있다. 또한, 컨트롤러가 펌웨어가 손상되거나 단절되었다고 탐지하면, 컨트롤러는 펌웨어를 업데이트하는 동안 프로세서를 정지시킬 수 있다.

*대 표 도* - 도2



## *특허청구의 범위*

**청구항 1**

오퍼레이팅 시스템을 실행하는 프로세서;

구성 데이터(configuration data)를 포함하는 제1 영역 및 제2 영역을 가지는 펌웨어 - 상기 제1 영역은 상기 제2 영역이 손상되었는지를 판단하기 위해 상기 제2 영역을 모니터링하는 로직을 포함함 - ; 및

업데이트된 구성 데이터를 요구하기 위해, 서버의 메모리에 저장된 데이터베이스로부터의 업데이트된 구성 데이터를 요구하는 메시지를 네트워크를 통해 서버에 보내고, 상기 네트워크를 통해 상기 업데이트된 구성 데이터를 수신하고, 상기 수신된 업데이트된 구성 데이터로 상기 제2 영역의 적어도 일부를 덮어쓰기 위한 펌웨어를 포함하는 컨트롤러 - 상기 메시지는 상기 프로세서 및 상기 펌웨어를 포함하는 연산 시스템의 식별을 포함함 - ;

를 포함하고,

상기 제1 영역은 구성 데이터의 업데이트가 필요하다고 표시하는, 선택된 시간인지 또는 시간의 기간이 만료되었는지를 판단하기 위해 클럭 신호를 모니터링하는 로직을 포함하고, 상기 제1 영역은, 상기 연산 시스템의 부팅이 개시되었는지를 상기 컨트롤러에 표시하기 위해, 사전정의된 통지 신호를 보내기 위한 하드웨어 요소, 및 상기 컨트롤러에 사전정의된 통신 소프트웨어 통지 메시지를 보내기 위한 소프트웨어 요소 중 하나를 활성화시키는 장치.

**청구항 2**

제1항에 있어서,

상기 제1 영역은 상기 구성 데이터에 따라 상기 프로세서의 프로그레스를 프로세서의 예상 응답과 비교하기 위해 클라이언트 칩셋을 모니터링하는 펌웨어를 포함하는 장치.

**청구항 3**

제1항에 있어서,

상기 제1 영역은 상기 제2 영역에 결합되고, 상기 제2 영역이 정지, 또는 단절되었는지를 판단하기 위해 제2 영역을 모니터링하는 로직을 포함하는 장치.

**청구항 4**

제3항에 있어서,

상기 제1 영역은 상기 제2 영역이 손상, 정지, 및 단절 중 하나에 해당할 때 상기 프로세서를 정지시키는 로직을 포함하는 장치.

**청구항 5**

제4항에 있어서,

상기 제1 영역은 상기 제2 영역이 손상, 정지 및 단절 중 하나에 해당하더라도 실행될 수 있는 회로 또는 펌웨어를 포함하는 장치.

**청구항 6**

삭제

**청구항 7**

제1항에 있어서,

상기 제1 영역은 소거 보호 메모리에 저장되고, 상기 메모리는 상기 제2 영역이 손상되었다는 것을 표시하기 위해 사전정의된 메시지를 컨트롤러에 보내는 하드-코딩된 시스템 파라미터를 포함하는 장치.

**청구항 8**

제7항에 있어서,

상기 컨트롤러는 내장된 마이크로-컨트롤러, 펌웨어, 및 회로 중 하나를 포함하는 장치.

**청구항 9**

제7항에 있어서,

상기 제1 영역은 상기 제2 영역의 체크섬-기반(checksum-based) 체크, 서명-기반 체크, 및 하드웨어-기반 체크 중 하나를 수행하기 위해 로직을 포함하는 장치.

**청구항 10**

제1항에 있어서,

상기 제1 영역은 상기 컨트롤러에 저장되는 장치.

**청구항 11**

삭제

**청구항 12**

삭제

**청구항 13**

삭제

**청구항 14**

제1항에 있어서,

상기 로직은 회로, 펌웨어, 소프트웨어 코드, 디지털 데이터, 및 소프트웨어 중 하나를 포함하는 장치.

**청구항 15**

연산 시스템의 펌웨어가 손상되었는지를 판단하기 위해 상기 연산 시스템을 모니터링하는 단계;

상기 펌웨어의 영역이 손상된 경우 상기 손상된 영역을 업데이트하기 위해 상기 연산 시스템으로부터, 네트워크를 통해 업데이트 데이터를 요구하는 단계;

상기 시스템의 프로그레스를 모니터링하는 단계;

상기 펌웨어에 저장된 장치에 대한 구성을 이용하여 판단된 상기 장치로부터의 예상 응답과 상기 연산 시스템의 상기 장치에 의해 수행되는 기능(function)의 차이를 탐지하는 단계를 포함하는, 상기 펌웨어에 따른 예상 응답과 상기 프로그레스 사이의 불일치를 탐지하는 단계;

상기 펌웨어가 손상되었다고 판단되면 상기 연산 시스템의 프로세서가 프로세싱을 정지하도록 하는 단계; 및

(1)상기 펌웨어가 단절 또는 정지되거나 (2)상기 프로세서가 중단되면, 손상된 펌웨어를 업데이트하는 단계

를 포함하고,

상기 업데이트하는 단계는 선택된 시간에 또는 선택된 시간의 기간 후에, 상기 펌웨어에 대한 펌웨어 업데이트 데이터를 요구하는 단계를 포함하고, 상기 업데이트 데이터를 요구하는 단계는, (1)상기 펌웨어가 단절 또는 정지되거나 (2)상기 프로세서가 중단된 후에 업데이트 데이터를 요구하는 단계를 포함하는 머신 구현 방법.

**청구항 16**

제15항에 있어서,

상기 모니터링하는 단계는 체크섬-기반 체크, 서명-기반 체크, 및 하드웨어-기반 체크 중 하나를 수행하는 단계를 포함하는 머신 구현 방법.

**청구항 17**

제15항에 있어서,

모니터링하는 단계는 상기 펌웨어가 단절 또는 정지되면 상기 펌웨어가 손상되었다고 판단하는 단계를 포함하는 머신 구현 방법.

**청구항 18**

삭제

**청구항 19**

삭제

**청구항 20**

삭제

**청구항 21**

삭제

**청구항 22**

삭제

**청구항 23**

네트워크 접속을 통해 펌웨어 업데이트 데이터를 수신하는 단계; 및

상기 펌웨어 업데이트 데이터로 연산 시스템의 펌웨어를 업데이트하는 단계 - 상기 펌웨어는 제1 영역 및 제2 영역을 포함하고, 상기 업데이트하는 단계는 상기 펌웨어의 상기 제1 영역에 의해 상기 네트워크 접속을 통해 요구된 업데이트된 제2 영역 데이터로 상기 펌웨어의 상기 제2 영역을 업데이트하는 단계를 포함함 -

를 포함하고,

상기 수신하는 단계 및 업데이트하는 단계는 상기 연산 시스템의 상기 펌웨어가 정지되면 행해지고,

상기 업데이트하는 단계는 선택된 시간에 또는 선택된 시간의 기간 후에 상기 펌웨어에 대한 펌웨어 업데이트 데이터를 요구하는 단계를 포함하고, 상기 선택된 시간 또는 상기 선택된 시간의 기간은 관리자에 의해 선택되거나 상기 연산 시스템의 일부를 제조하는 동안 선택되고,

상기 업데이트하는 단계는, 상기 펌웨어의 피쳐 또는 프로세스에서 오류를 제거하거나, 상기 펌웨어에 피쳐 또는 프로세스를 더하거나, 상기 펌웨어의 피쳐 또는 프로세스를 향상시키는 것 중 하나를 위해 상기 펌웨어를 업데이트하는 단계를 포함하는 머신 구현 방법.

**청구항 24**

삭제

**청구항 25**

제23항에 있어서,

상기 업데이트하는 단계는

상기 네트워크를 통해 요구된 업데이트된 제1 영역 데이터로 상기 펌웨어의 제1 영역을 업데이트하는 단계; 및

상기 제1 영역에 의해 요구된 업데이트된 제2 영역 데이터로 상기 펌웨어의 제2 영역을 업데이트하는 단계

를 포함하는 머신 구현 방법.

**청구항 26**

제23항에 있어서,

상기 업데이트하는 단계는

상기 연산 시스템에 대한 고유 식별을 이용하여 펌웨어 업데이트 데이터를 요구하는 단계;

서버로부터 상기 네트워크를 통해 펌웨어 업데이트 데이터를 수신하는 단계;

상기 펌웨어 업데이트 데이터를 상기 펌웨어에 반환하거나 상기 펌웨어 업데이트 데이터로 상기 펌웨어를 덮어쓰는 단계

를 포함하는 머신 구현 방법.

**청구항 27**

삭제

**청구항 28**

제23항에 있어서,

상기 수신하는 단계는 보안 서버 세션을 사용하는 단계 및 암호화된 데이터를 사용하는 단계 중 하나를 포함하는 머신 구현 방법.

**청구항 29**

삭제

**청구항 30**

삭제

## *명 세 서*

### *기 술 분 야*

<1> 본 발명은 시스템 펌웨어(firmware)를 업데이트 및 복구하는 분야에 관한 것이다.

### *배 경 기 술*

<2> 연산 시스템(computing system)과 같은 전자 장치를 초기화하거나 사용하는 동안, 펌웨어 "프로그램"은 장치 또는 시스템 하드웨어를 탐지, 테스트, 초기화 및 모니터링하는데 종종 사용된다. 예를 들어, 퍼스널 컴퓨터(PC)가 켜질 때, 통상적으로 BIOS 칩으로 불리우는, PC 마더보드 상의 리드-온리 메모리(ROM) 칩에 저장 또는 프로그램밍된 소프트웨어인 베이직 입력/출력 시스템(BIOS) 프로그램(예를 들어, 펌웨어)이 실행된다. BIOS가 실행됨과 동시에 즉각적으로 보드 상의 장치(예를 들어, 프로세서, 메모리, 비디오 카드 등)의 모두를 식별하고, 각 장치가 제대로 작동하는지를 판단하기 위해 파워-온 자기-테스트(Power-on self-test; POST)를 실행한다. 모든 장치가 POST 테스트를 통과하면 BIOS는 각 장치를 초기화하고, 하드 드라이브, CD-ROM 드라이브 및 플로피 드라이브를 탐지한다. 그리고 나서 BIOS는 오퍼레이팅 시스템 소프트웨어를 로딩하기 위해 필요한 파일을 찾기 위해 제 1 부팅 장치(일반적으로 하드 드라이브 또는 플로피 디스크)를 검색한다. BIOS는 키보드 및 디스플레이 컨트롤러와 같은 주변 장치에 대한 저-레벨 입력/출력(I/O)을 다룰 수 있다. 또한, BIOS는 오퍼레이팅 시스템이 끝날 때까지 시스템 하드웨어를 탐지, 테스트, 초기화 및 모니터링할 것이다. 그리고 나서, BIOS는 시스템의 제어를 오퍼레이팅 시스템 소프트웨어에 전달할 수 있다. 임의의 장치가 POST를 통과하지 못하면, 문제가 존재한다는 것을 알리기 위해 스크린 상에 오류 메세지가 디스플레이되거나 PC 스피커를 통해 연속적인 "경고음(beeps)"이 날 것이다. 경고음의 시퀀스(경고음 코드)는 나타난 문제의 타입을 식별하는데 사용될 수 있다.

<3> 그러나, BIOS 또는 펌웨어는 통상적으로 ROM 칩의 한 부분에 "구워지고(burned)" ROM 칩의 제2 부분에 쓰여지거나 프로그래밍된다. "구워진(burned)" 부분은 하드웨어 메커니즘(예를 들어, 자외선 투사에 의해)에 의해 "플래시(flash)" 소거될 수 있지만, 제2 부분은 소프트웨어 메커니즘에 의해 소거되거나 쓰여질 수 있다. 결과적으로, BIOS 또는 펌웨어는 "바이러스(virus)", "웜(worm)" 또는 "해커(hacker)"에 의해 손상되거나, 부정확해지거나, 바람직하지 않게 변경될 수 있다. 게다가, 연산 시스템의 사용자가 뜻하지 않게 BIOS를 바람직하지 않은

형태로 변경할 수 있다. 따라서, BIOS 또는 펌웨어를 정확하게 복구하고 업데이트할 수 있는 것이 중요하다.

## *발명의 상세한 설명*

<8> 도 1은 네트워크를 통해 펌웨어를 셀프-모니터링하고 업데이트하는 시스템의 블록도이다. 도 1은 통신 링크(162)에 의해 네트워크(170)에 연결된 연산 시스템(102), 및 통신 링크(182)에 의해 네트워크(170)에 연결된 서버(180)를 포함하는 연산 환경(computing environment, 100)을 나타낸다. 연산 장치(102)는 클라이언트 컴퓨터, 서버 컴퓨터, 데스크탑 컴퓨터, 노트북 컴퓨터, 퍼스널 디지털 보조기(PDA), 휴대폰 또는 임의의 다른 디지털 프로세서 또는 오퍼레이팅 시스템 및 펌웨어를 가지는 프로세싱 시스템이 될 수 있다.

<9> 도 1에 보이는 바와 같이, 연산 시스템(102)은 네트워크 인터페이스(160)를 통해 통신 링크(162)에 연결된 버스(150)를 포함한다. 버스(150)는 컴퓨터의 수많은 요소와 인터페이스하기 위한 마더보드 상의 버스와 같은 컴퓨터 버스가 될 수 있다. 네트워크 인터페이스(160)는 로컬 영역 네트워크(LAN), 인트라넷, 인터넷, 또는 또 다른 전자 장치 또는 연산 시스템 통신 지원 네트워크(computing system communication supporting network)에 연결되는 네트워크 인터페이스가 될 수 있다. 통신 링크(162) 및 네트워크(170)는 상기 네트워크에 대해 통신을 지원한다. 특히, 예를 들어, 네트워크(170)는 지역 영역 네트워크(LAN), 인트라넷, 인터넷이 될 수 있다. 네트워크(170)는 전자 장치 및/또는 연산 시스템 간의 통신을 제공하기 위해 하나 이상의 네트워크 및/또는 기술 타입을 포함하는 것 역시 고려할 수 있다. 특히, 네트워크(170)는 무선 통신, 셀 방식 통신, LAN 통신, 및 인터넷을 포함할 수 있다. 통신 링크(182)는 상기 기재된 통신 링크(162)와 유사한 링크가 될 수 있다. 서버(180)는 본 명세서에 기재된 기능을 제공하기 위한 네트워크 서버, 연산 시스템(102)과 유사한 연산 시스템, 또는 그 밖의 컴퓨터 서버가 될 수 있다.

<10> 연산 시스템(102)은 또한 버스(150)에 연결된 키보드/마우스 인터페이스(152)를 포함하여 도시되어 있다. 키보드 마우스 인터페이스(152)는 입력을 연산 시스템에 제공하기 위해 키보드 및/또는 마우스를 연산 시스템(102)에 연결시키는 인터페이스가 될 수 있다. 또한, 도시된 바와 같이, 모니터 인터페이스(154)가 버스(150)에 연결된다. 모니터 인터페이스(154)는 연산 시스템(102)에 연결된 스크린 또는 모니터에 대해 인터페이스 또는 적당한 신호를 제공한다. 예를 들어, 모니터 인터페이스(154)는 비디오 또는 디스플레이 카드가 될 수 있다.

<11> 도 1은 버스(150)에 연결된 메모리(142)를 또한 보여준다. 메모리(142)는 랜덤 액서스 메모리(RAM), 메모리 저장소(memory storage), 유니버설 시리얼 버스(USB) 메모리, 하드 드라이브, CD-ROM, 광 디스크, 및/또는 플로피 디스크를 나타낼 수 있다. 메모리(142)는 명령어들을 가지는 미디어와 같은 머신 액세스 가능 미디어가 될 수 있고, 프로세서에 의해 실행되는 경우 연산 시스템(102)으로 하여금, 네트워크(170)를 경유해 다른 연산 시스템 및/또는 서버(180)와 통신하는 것과 같은 본 명세서에 기재된 컴퓨터-관련 작업을 수행하도록 한다.

<12> 또한 표시된 바와 같이, 프로세서(140)가 버스(150)에 연결된다. 프로세서(140)는 중앙 처리 장치(CPU), 디지털 신호 프로세서, 또는 다른 프로세서가 될 수 있다. 프로세서(140)는 메모리(142)에 의해 제공되거나 메모리에 저장된 명령어를 처리하는 프로세서가 될 수 있는데, 이를 테면 연산 시스템(102)으로 하여금 네트워크(170)를 경유해 다른 연산 시스템 및/또는 서버(180)와 통신하는 것과 같은 본 명세서에 기재된 컴퓨터-관련 작업을 수행하도록 한다. 프로세서(140)는 메모리(142)에 저장되거나 메모리(142)로부터 액세스 가능한 오퍼레이팅 시스템을 실행하거나 작동할 수 있다. 따라서, 연산 시스템(102)은 네트워크(170)를 경유하여 서버(180)에 액세스할 수 있는 클라이언트 컴퓨터 또는 클라이언트 시스템이 될 수 있다.

<13> 도 1은 또한 버스(150)에 연결된 펌웨어(110)를 도시한다. 펌웨어(110)는 부트 프로그램 또는 연산 시스템(102)의 오퍼레이팅 시스템을 테스트, 초기화 및 로딩하기 위한 컴퓨터 베이직 입력/출력 시스템(BIOS), 클라이언트 시스템 펌웨어(CSF) 또는 다른 명령어가 될 수 있다. 펌웨어(110)는 리드-온리 메모리(ROM), 비-휘발성 메모리, 소거 프로그램 가능 ROM(EPROM), 전기적 소거 프로그램 가능 ROM(EEPROM), 또는 플래시 메모리와 같은 메모리에 저장될 수 있다. 예를 들어, 펌웨어(110)는 연산 시스템(102)의 전원이 제거되거나 꺼지는 경우(즉, 하드 오프(hard off)의 경우)에도 소거되지 않는 메모리에 저장될 수 있다. 또한, 메모리를 업데이트, 복구, 또는 차후 데이터로 재작성하기 위해, 메모리를 자외선에 노출하거나 방사시키는 것과 같은 방법, 이를테면 메모리에 "굽기(burning)"에 의한 방식을 통해 펌웨어(110)는 소거 가능 메모리에 저장될 수 있다. 펌웨어(110)는 다른 프로세스, 펌웨어, 소프트웨어 및/또는 하드웨어에 의해 업데이트될 수 있는 점도 고려할 수 있다.

<14> 펌웨어가 손상되면(corrupted) BIOS 및 펌웨어(110)와 같은 펌웨어의 업데이트 또는 복구가 필요할 수 있다. 펌웨어는 연산 시스템의 장치들에 비하여 부정확한 구성 데이터(configuration data)를 가지거나; 오퍼레이팅 시스템 또는 프로세서로 하여금 멈추거나(halted), 단절되거나(hung up), 소프트-오프(soft-off) 상태로 되게

한다면 손상된 것으로 간주될 수 있다. 예를 들어, 멈추게 되는 경우는 하드웨어 정지 또는 다른 고장이 오퍼레이팅 시스템 및/또는 프로세서로 하여금 펌웨어, 소프트웨어, 또는 오퍼레이팅 시스템 명령어들의 프로세싱을 중지시키도록 하는 때가 될 수 있다. 마찬가지로, 단절되는 경우는 소프트웨어 루핑(looping) 문제 또는 다른 고장이 오퍼레이팅 시스템 및/또는 프로세서로 하여금 펌웨어, 소프트웨어, 또는 오퍼레이팅 시스템 명령어의 프로세싱을 중지시키도록 하는 때가 될 수 있다. 마지막으로, 소프트-오프는, 연산 시스템에 대하여 전원이 켜져 있지만, 전원 절약 모드, 또는 또 다른 하드웨어 또는 소프트웨어 모드에 있는 연산 시스템으로 인하여 오퍼레이팅 시스템 및/또는 프로세서는 펌웨어, 소프트웨어, 또는 오퍼레이팅 시스템 명령어의 프로세싱을 중단하는 때가 될 수 있다.

<15> 펌웨어는 피쳐(feature) 또는 프로세스에서 "버그(bug)" 또는 에러를 제거하거나, 피쳐 또는 프로세스를 추가하거나, 또는 펌웨어의 피쳐 또는 프로세스를 향상시키기 위해(예를 들면, 새로운 장치의 구성을 추가하기 위해) 업데이트 되거나 복구될 수 있다. 추가로, 펌웨어가 "바이러스", "웜", 또는 "해커"에 의해 손상되거나 변경된다면 펌웨어를 업데이트 또는 복구하는 것이 필요할 수 있다. 어떤 경우는 펌웨어가 계속 정확하도록 보장하기 위해, 또는 상기 기술된 바와 같이 펌웨어에 업데이트를 제공하기 위해, 지정된 시간에서 또는 주기적으로 펌웨어를 자동으로 업데이트하는 것이 바람직할 수 있다.

<16> 도 1은 또한 버스(150)에 연결된 컨트롤러(190)를 보여준다. 컨트롤러는 컴퓨터 칩, 디지털 신호 프로세서, 내장된 마이크로-컨트롤러, 펌웨어, 회로, 컴퓨터 하드웨어, 및/또는 여기에 저장된 컴퓨터 소프트웨어를 포함할 수 있다. 특히, 컨트롤러(190)는 펌웨어 FW를 가지는 것으로 도시되어 있다. 컨트롤러(190)는 본 명세서에 기재된 일을 수행하기 위해 코드, 프로그래밍, 펌웨어 FW, 데이터, 지능(intelligence), 및/또는 컴퓨터 소프트웨어와 같은 "로직(logic)"을 가지는 지능형 네트워크 컨트롤러(INC)가 될 수 있다. 일반적으로, 본 명세서에 기재된 코드, 프로그래밍, 데이터, 지능, 컴퓨터 소프트웨어, 펌웨어, 하드웨어, 및/또는 컴퓨터 하드웨어는 "로직"으로 정의될 수 있는데, 이러한 로직은 그 로직에 관해 기술된 기능을 수행할 능력을 가지고 있다.

<17> 예를 들어, 컨트롤러(190)는 펌웨어(110)를 모니터링하고, 서버(180)와 통신하는 로직을 포함할 수 있다. 좀 더 구체적으로, 서버(180)로부터 펌웨어 이미지를 검색하여 펌웨어(110)를 업데이트하기 위해, 컨트롤러(190)는 네트워크(170)를 통해 서버(180) 및 펌웨어(110)와 통신하는 로직을 포함할 수 있다. 컨트롤러(190)의 로직은 네트워크를 통해 펌웨어(110)에 대한 업데이트된 구성 또는 펌웨어 데이터를 요구하고 수신하는 능력을 가질 수 있다. 예를 들어, 컨트롤러(190)는 보안 서버 세션(secure server session) 또는 암호화된 데이터를 이용하여 네트워크(170)를 통해 서버(180)와 안전하게 통신할 수 있는 능력을 또한 가질 수 있다. 구체적으로, 통신 링크(162 및 182), 및 네트워크(170)를 통한 연산 시스템(102)과 서버(180) 간의 통신은 거래 계층(transactional layer; TSL) 보안 서버세션과 같은 보안 서버 세션이 될 수 있고/있거나 Ribest Shamir Adleman(RSA) 암호 데이터, 또는 또 다른 공개/전용 키 암호 데이터와 같은 암호 데이터를 사용할 수 있다. 컨트롤러(190)는 수신된 업데이트된 구성 또는 펌웨어 데이터를 펌웨어(110)에 반환하거나, 수신된 업데이트된 구성 또는 펌웨어 데이터를 펌웨어(110)에 덮어쓰는 로직을 또한 가질 수 있다.

<18> 컨트롤러(190)는 영역(120)이 실행되고 있거나 손상된 때 기능성 프리젠스(functional presence)를 가지기 위해 충분한 로직을 가질 수 있다. 실시예에 따르면, 컨트롤러(190)는 펌웨어(110)와 독립적으로 실행하는 로직 또는 펌웨어를 포함함으로써, 영역(120)이 손상되거나 단절된 때; 프로세서(140)가 보류되거나 정지된 때; 연산 시스템(102)의 오퍼레이팅 시스템이 단절되거나, 정지되거나, 또는 소프트-오프되는 때; 및/또는 연산 시스템(102)이 단절되거나, 정지되거나, 또는 소프트-오프된 때에도 컨트롤러(190)는 완전하게 작동한다. 예를 들어, 프로세서(140)는 하드웨어 또는 소프트웨어 고장으로 인해 프로세서가 펌웨어, 소프트웨어, 또는 오퍼레이팅 시스템 명령어의 프로세싱을 중단하는 경우 보류 또는 정지된다. 따라서, 컨트롤러(190)는 프로세서(140), 또는 연산 시스템(102)의 오퍼레이팅 시스템이 단절되거나, 정지되거나, 또는 소프트-오프되는 경우에도 완전히 작동할 수 있다.

<19> 또한, 컨트롤러(190)는 펌웨어의 일부가 손상되었다는 표시를 수신하기 위해 그리고 서버(180)로부터 검색되는 원본 또는 업데이트된 펌웨어에 대한 요구를 수신하기 위해, 펌웨어(110)와 통신하는 충분한 로직을 가질 수 있다. 컨트롤러(190)는 펌웨어 또는 메모리, 수정 정보, 배포 데이터 등을 가지거나 저장할 수 있다.

<20> 컨트롤러(190), 메모리(142), 및/또는 펌웨어(110)는 펌웨어(110), 연산 시스템(102)의 오퍼레이팅 시스템, 및/또는 연산 시스템(102) 자체의 식별 정보, 수정 정보, 배포 데이터 등을 얻기 위해, 로직, 정보, 및/또는 메커니즘을 가질 수 있다. 예를 들어, 연산 시스템(102)의 식별은 소정의 회사, 소정의 조직, 또는 연산 시스템(102)이 위치한 위치 또는 빌딩에 있는 유사한 연산 시스템 사이에서(예를 들어, 다른 클라이언트 시스템 사이

에서) 연산 시스템(102)의 고유 식별이 될 수 있다. 구체적으로, 컨트롤러(190)는, 고유 식별과 같은 연산 시스템 식별에 기초하여 연산 시스템(102)을 식별하기 위해 연산 시스템의 다른 요소와 통신하는(예를 들어, 프로세서(140) 및/또는 펌웨어(110)와 통신하는) 로직을 포함할 수 있다.

<21> 또한, 컨트롤러(190)는 펌웨어에 대한 요구 및 연산 시스템(102)의 식별을 네트워크(170)를 통해 서버(180)와 통신하고, 상기 요구와 관련된 응답을 서버(180)로부터 수신하고, 수신된 응답을 펌웨어 및/또는 연산 시스템(102)의 다른 요소에 보내기 위한 로직을 포함할 수 있다. 예를 들어, 컨트롤러(190)는 영역(120)으로부터 "코어 손상(core corrupted)" 메시지, 업데이트 데이터 요구 메시지(request update data massage), 또는 펌웨어 업데이트 데이터 요구 메시지(request firmware update data massage)를 수신할 수 있다. 예를 들어, "코어"는 펌웨어(110) 또는 이의 일부 영역으로 정의될 수 있다. 그에 응답하여, 컨트롤러(190)는 서버(180)에 "코어 손상(core corrupted)" 메시지, 업데이트 데이터 요구 메시지, 또는 펌웨어 업데이트 데이터 요구 메시지를 보낼 수 있다. 컨트롤러(190)는 그리고 나서 서버(180)로부터 펌웨어 업데이트 데이터를 기다리고 수신하며, 상기 펌웨어 업데이트 데이터를 펌웨어(110)에 제공한다. 또한, 컨트롤러(190)는 서버(180)로부터 수신한 펌웨어 업데이트 데이터로 영역(120) 및/또는 영역(130)을 덮어쓰기 위해 서버(180)에 대한 통과 인터페이스(pass-through interface)를 제공하거나 또는 덮어쓸 수 있는 로직을 포함할 수 있다.

<22> 추가로, 연산 시스템(102)은 칩셋(125)을 포함한다. 칩셋(125)은 칩셋(125)에 관해 본 명세서에 기재된 기능을 수행하는 능력을 가진 컴퓨터 칩, 디지털 신호 프로세서, 펌웨어, 회로, 컴퓨터 하드웨어 및/또는 여기에 저장된 컴퓨터 소프트웨어, 레지스터, 버퍼, 컴퓨터 로직, 게이트 등을 포함할 수 있다. 일부 경우에서, 컨트롤러(190)는 펌웨어(110)와 프로세서(140) 사이의 통신을 모니터링하기 위해 칩셋(125)을 모니터링함으로써 프로세서(140)의 프로그레스(progress)를 모니터링할 수 있다. 따라서, 칩셋(125)은 컨트롤러(190)로 하여금 프로세서(140)의 의한 펌웨어(110)의 액세스를 모니터링하도록 하며 연산 시스템(102)이 부팅되는 동안 오류가 존재하는지 탐지하기 위해 프로세서(140)로부터 받은 데이터를 펌웨어(110)에 반환한다. 구체적으로, 칩셋(125)은 컨트롤러(190)로 하여금 펌웨어(110) 및 프로세서(140) 사이의 통신을 모니터링하도록 하여, 펌웨어(110)의 디폴트(default) 주소들의 액세스에 의해 시스템(102)의 부팅을 시작하게 하고, 칩셋(125)를 통과하는 데이터를 반환한다.

<23> 도 1은 통신 링크(162 및 182) 및 네트워크(170)를 통해 연산 시스템(102)에 연결된 서버(180)를 보여준다. 서버(180)는 사무, 회사, 위치, 또는 조직에서의 컴퓨터의 펌웨어 구성을 제어하기 위해, 정보 기술(IT) 관리자에 의해 모니터링되고 제어되는 서버 콘솔과 같은 컴퓨터 "콘솔(console)"이 될 수 있다. 예를 들어, 펌웨어 업데이트를 사무, 회사, 위치, 또는 조직의 컴퓨터에 제공하기 위해 IT 관리자는, 서버(180)에서 일하거나 서버에 액세스하는 사람, 또는 서버(180)에서 실행하거나 서버에 액세스하는 자동화 기기가 될 수 있다. 서버(180)는 연산 시스템(102)과 같은 연산 시스템에 대한 구성 데이터 및 펌웨어 업데이트를 포함하는 메모리 저장 데이터베이스를 가질 수 있다. 또한, 서버(180)는 오퍼레이팅 시스템 이미지를 가질 수 있다. 서버(180)의 펌웨어 업데이트 및 오퍼레이팅 시스템 이미지는 서버의 데이터베이스를 색인함으로써 액세스될 수 있다. 다른 대안으로서, 펌웨어 업데이트 및 오퍼레이팅 이미지는 관리자(예를 들어, 서버(180)에 액세스하는 사람 또는 서버(180)에서 실행하거나 서버에 액세스하는 자동화 기기)에 의해 서버(180)로부터 제공될 수 있는데, 이를테면 정보 기술(IT) 관리자는 서버(180) 내의 데이터베이스로부터 수동으로 펌웨어 및/또는 오퍼레이팅 시스템 이미지를 선택하거나 또는 다른 방법으로 네트워크(170)를 통한 유포를 위해 펌웨어 및/또는 오퍼레이팅 시스템 이미지를 서버(180)에 로딩한다.

<24> 일 실시예에서, 서버(180)는 식별 정보 및 관련 지원 데이터(예를 들어, 펌웨어 데이터, 펌웨어 업데이트, 펌웨어 이미지, 오퍼레이팅 시스템 이미지 등)의 데이터 베이스를 포함할 수 있다. 적당한 식별 정보는 클라이언트 시스템 식별(예를 들어, 연산 시스템(102))(예를 들어, 조직의 다른 클라이언트 시스템 사이에서 고유 식별이 될 수 있는 식별), 오퍼레이팅 시스템 버젼, 프로세서 버젼, 펌웨어 수정 정보, 펌웨어 배포 일자 등을 포함한다. 구체적으로, 서버(180)는 컨트롤러(190)로부터 수신된 클라이언트 시스템 식별에 대응하는 데이터를 지원하기 위해 액세스를 순람(lookup), 추적, 및 송신하거나 제공한다(예를 들어, 다양한 클라이언트 시스템 식별, 펌웨어 수정, 및/또는 펌웨어 배포 일자에 대해). 일부 경우에서, 서버(180)는 IT 관리자가 액세스할 수 있어서 IT 관리자의 행동은 지원 데이터로의 액세스를 순람, 추적, 및 송신하거나 제공할 수 있다.

<25> 도 1은 영역(120) 및 영역(130)을 포함하는 펌웨어(110)를 보여준다. 영역(120) 및 영역(130)은 펌웨어(110)의 전체 펌웨어의 일부 또는 모든 영역이 될 수 있다. 실시예에 따르면, 영역(120) 및 영역(130)을 포함하여 함께 "코어 시스템" 펌웨어로 부를 수 있으며, 영역(120)은 파트 1 또는 코어 시스템 펌웨어의 "부트 블록(boot block)"이 되며, 영역(130)은 파트 2 또는 코어 시스템 펌웨어의 "코어"가 된다. 예를 들어, 영역(120)은 하드

웨어 기반 보호 및/또는 소프트웨어 기반 보호에 의해 소거 보호되는 펌웨어의 제1 영역이 될 수 있다. 영역(120)은 영역(120)을 소거하기 위해 특정 하드웨어 메커니즘(예를 들어, 자외선 투사)을 사용하지 않는 한, 의도적 혹은 우연적인 소거로부터 소거 보호되는 펌웨어(110)의 작은 부분 또는 소거 보호 부분이 될 수 있다. 실시예에 따르면, 영역(120)은 ROM, EPROM, 또는 EEPROM 칩의 일부 영역에 "구워진" 데이터를 나타낼 수 있다. 따라서, 영역(120), 즉 "구워진" 영역은 하드웨어 메커니즘에 의해(예를 들어, 자외선 투사에 의해) "플래시(flash)" 소거될 수 있다.

<26> 추가로, 영역(120)은 시스템 플랫폼 상, 예를 들어, 연산 시스템(120) 상의 컨트롤러(190)의 존재를 발견하고 인지하는 로직을 포함할 수 있다. 예를 들어, 영역(120)은 시스템 플랫폼 상의 컨트롤러(190)의 하드-코딩된 위치 및/또는 런-타임(run-time) 발견을 인식할 수 있다. 더욱이, 영역(120)은 이를테면 영역(130)이 손상되었는지를 판단하기 위한 체크섬(checksum)-기반 체크, 서명-기반 체크, 및/또는 하드웨어-기반 체크를 사용하여, 펌웨어(110) 내에 손상된 펌웨어를 발견하기 위해 충분한 로직을 포함할 수 있다. 게다가, 영역(120)은 컨트롤러(190)와 통신하기 위한 로직을 포함할 수 있다. 따라서, 영역(120)은 영역(120) 및/또는 영역(130)이 손상되거나, 업데이트가 필요한 경우 컨트롤러(190)와 통신할 수 있다. 예를 들어, 영역(120)은 컨트롤러(190)로부터 영역(120) 및/또는 영역(130)에 대한 원본 또는 업데이트된 펌웨어를 요구할 수 있다. 영역(120)은 요구에 대응하는 응답, 원본 펌웨어, 업데이트 펌웨어, 및/또는 업데이트 데이터를 수신하기 위해 충분한 로직을 포함하며, 이와 함께 이를 테면, 영역(120) 및/또는 영역(130)을 펌웨어 또는 데이터로 업데이트함으로써 적절한 조치를 취한다.

<27> 펌웨어(110)는 소거가능한 펌웨어 영역과 같은 영역(130)을 또한 포함한다. 실시예에 따르면, 영역(130)은 메모리, ROM, EPROM, 또는 EEPROM 칩의 영역에 쓰여지거나 저장된 데이터를 나타낼 수 있다. 예를 들어, 영역(130)은 영역(120)이 프로그래밍되어있는 ROM 칩의 제2 부분에 쓰여지거나 프로그래밍되지만, 영역(120)의 하드웨어 또는 소프트웨어 소거 보호를 가지지 않을 수 있다. 따라서, 영역(130)은 소프트웨어 메커니즘에 의해 삭제되거나 덮어쓰여질 수 있다. 영역(130)은 영역(120)보다 데이터 사이즈가 훨씬 크다. 영역(130)은 영역(120)에 의해 또는 이로부터 발생한 삭제, 업데이트, 복구를 위해 설계될 수 있다. 영역(120 및 130)은 연산 시스템(102)에 대한 구성 데이터를 포함할 수 있다. 예를 들어, 구성 데이터는 키보드/마우스 인터페이스(152) 및 모니터 인터페이스(154)와 같은 연산 시스템(102)의 장치 또는 이에 부착된 장치의 구성을 포함할 수 있다. 펌웨어(110)로부터의 데이터는, 상기 언급된 장치에 대한 동작 파라미터를 규정하기 위해 연산 시스템(102)의 구성 레지스터를 로딩하는데 사용될 수 있다.

<28> 따라서, 실시예에 따르면, 연산 환경(100)은 연산 시스템(102)의 오퍼레이팅 시스템이 적절하게 작동하고 있는지 아닌지 간에, 단절된 때, 손상된 때, 또는 특정 시간에 펌웨어(110)가 업데이트되도록 한다. 예를 들어, 펌웨어(110)는 클라이언트 시스템인 연산 시스템(102)에 존재하는 클라이언트 시스템 펌웨어(CSF)가 될 수 있고, CSF는 CSF가 손상되었는지를 발견하기 위해 이에 내장된 로직을 가지며, 손상된 CSF를 "자체-복구" 또는 업데이트하는 능력을 가진다.

<29> 일 실시예에서, 컨트롤러(190)는 지능형 네트워킹 컨트롤러(INC)가 될 수 있는데, 이는, 이를테면 손상된 펌웨어를 탐지하고, 연산 시스템(102)을(예를 들어, 네트워크(170)을 통해) 서버(180)와 상호 작용할 수 있는 상태가 되도록 지원하는 로직을 가지는 컨트롤러이다. 이 로직은 컨트롤러(190)로 하여금 펌웨어(110)의 전체 또는 일부(예를 들면, 영역(120) 및/또는 영역(130))을 최초로 기재, 업데이트, 교정, 셀프-치료, 회복, 및 셀프-업데이트한 것에 대한 원본 또는 업데이트된 펌웨어(예를 들면, 코드, 연산 시스템(102)에 대한 펌웨어 기능 및/또는 장치 구성 정보를 제공하는 프로그래밍 또는 소프트웨어를 포함하는 데이터 또는 이미지)를 검색하도록 한다.

<30> 마찬가지로, 연산 시스템(102)에 있어서, 영역(120)이 컨트롤러(190) 및 (예를 들어, 네트워크(170)을 통해) 서버(180)와 상호작용할 수 있는 상태가 되도록 하기 위해, 영역(120)은 충분한 로직 기능을 포함하는 CSF의 한 부분 또는 영역이 될 수 있다. 여기서, 로직은 영역(120)으로 하여금 펌웨어(110)의 영역(예를 들면, 영역(130))을 최초로 기재, 업데이트, 교정, 셀프-치료, 회복, 및 셀프-업데이트한 것에 대한 원본 또는 업데이트된 펌웨어를 회복하도록 한다. 영역(120)의 소거 보호는 영역(130)의 복구(예를 들어, 컨트롤러(190) 및 서버(180)를 사용)를 가능하게 하는 로직을 보존하기 위해 구성되거나 내장될 수 있다.

<31> 유사하게, 영역(130)은 연산 시스템(102)의 구성요소 또는 장치의 구성 데이터를 가지는 CSF의 비-안전 부분 또는 영역이 될 수 있다. 예를 들어, 영역(130)은 영역(120)만큼 안전하지 않은 펌웨어의 주된 부분이 될 수 있는데, 영역(130)은 소거로부터 영역(130)을 보호하기 위해, 영역(120)을 위한 것과 유사한 내장된 하드웨어 또

는 소프트웨어 메커니즘을 가지고 있지 않기 때문이다. 따라서, 영역(130)은 영역(130)의 의도적인 삭제 또는 필드 업그레이드를 허용하는데 유연할 수 있다. 그러나 이러한 유연성은 영역(130)을 우연한 삭제에 취약하게 만들 수 있다. 따라서, 만약 영역(130)이 우연히 삭제되면, 영역(120)은 영역(130)의 회복을 도와줄 수 있다.

<32> 서버(180)는 연산 시스템(102)과 같은 클라이언트 시스템이 적절한 펌웨어 및/또는 오퍼레이팅 시스템 이미지를 받기 위해 통신하는 콘솔이 될 수 있다. 펌웨어 및/또는 오퍼레이팅 시스템 이미지는 서버(180)에 저장된 데이터베이스를 색인하여 서버(180)로부터 자동으로 선택될 수 있다. 다른 대안으로서, 정보 기술(IT) 관리자는 서버(180) 상의 데이터베이스로부터 정확한 펌웨어 및/또는 오퍼레이팅 시스템 이미지를 선택하고, 네트워크(170)를 통한 연산 시스템(102)으로의 선택된 펌웨어 및/또는 오퍼레이팅 시스템 이미지의 통신을 제어함으로써, 수동으로 펌웨어 및/또는 오퍼레이팅 시스템 이미지를 선택할 수 있다.

<33> 도 2는 네트워크를 통해 펌웨어를 업데이트하는 과정의 흐름도(200)이다. 블록(202)에서 시스템은 부팅되거나 리셋된다. 예를 들어, 블록(202)은 연산 시스템(102)과 같은 연산 시스템의 전원을 켜거나, 리셋하거나, 초기화하거나, 또는 개시하는 것에 해당할 수 있다. 다음으로, 블록(203)에서 컨트롤러는 부팅, 초기화, 펌웨어 실행, 오퍼레이팅 시스템 실행, 또는 프로세싱을 계속하는 시스템의 능력과 관계없이 개시되거나 초기화된다. 예를 들어, 블록(203)은 펌웨어(110), 영역(120) 및/또는 영역(130)이 손상되거나 단절되었는지 여부에 상관없이, 펌웨어를 실행하도록 초기화하고 구성하기 위해 개시되는 컨트롤러(190)에 해당할 수 있다. 마찬가지로, 컨트롤러(190)는 프로세서(140)가 중단되거나 멈추는지 여부에 상관없이; 연산 시스템(102)의 오퍼레이팅 시스템이 중단되거나, 단절되거나, 소프트-오프되는지 상관없이; 및/또는 연산 시스템(102)이 이와 다르게 중단되거나, 단절되거나, 소프트-오프되는지 여부에 상관없이 상기 기술된 기능을 수행할 수 있다.

<34> 블록(205)에서 컨트롤러(190)와 같은 컨트롤러는, 연산 시스템(102)의 스타트 업(start up)과 같은 클라이언트 시스템 스타트 업을 통지받는다. 블록(205)은 하드웨어 메커니즘(예를 들어, 하드웨어 또는 회로)에 의한 통지를 포함할 수 있는데, 이는 연산 시스템(102)의 부팅이 개시되었는지를 (예를 들어, 연산 시스템(102)의 전원이 켜지거나 연산 시스템(102)이 "리셋"된 후) 표시하기 위해 사전정의된(predefined) 신호를 컨트롤러에 보내는 통지이다. 또한, 컨트롤러(190)는 연산 시스템(102)의 부팅을 표시하기 위해 컨트롤러(190)에 보내진 사전정의된 통신 메시지와 같은 소프트웨어 메커니즘에 의한 스타트 업을 통지받는다. 통지는 하드웨어 인터페이스를 경유해 사전정의된 통지 신호를 송신하는 것, 또는 펌웨어(110) 및 컨트롤러(190) 사이의 통신 인터페이스를 경유하여 사전정의된 소프트웨어 메세지를 송신하는 것을 포함할 수 있다.

<35> 블록(210)에서 시스템은 펌웨어가 손상되었는지를 판단하기 위해 모니터링된다. 예를 들어, 컨트롤러(190)는 영역(120)이 손상되었는지 모니터링하거나, 판단하거나, 발견하기 위해 컨트롤러(190)의 펌웨어에 내장된 방법 또는 로직을 포함할 수 있다. 일부 실시예에서, 컨트롤러(190)는 하드웨어 칩셋을 이용하여 연산 시스템(102)의 부팅, 초기화 및/또는 펌웨어 실행의 프로그레스를 모니터링 할 수 있다. 구체적으로, 컨트롤러(190)는 영역(190)이 손상되거나 단절되었는지를 판단하기 위해 부팅 또는 초기화 동안 영역(120)의 실행을 모니터링할 수 있다. 일부 실시예에서, 컨트롤러(190)는 프로세서(140) 및 펌웨어(110) 사이의 칩셋을 모니터링함으로써 프로세서(140)의 프로그레스를 모니터링할 수 있다. 연산 시스템(102)이 부팅될 때, 프로세서(140)는 펌웨어(110) 내의 디폴트 어드레스(예를 들어, 영역(120)의 어드레스)에 액세스할 수 있고 부팅이 성공적인지 여부에 따라 펌웨어(110)에 데이터를 반환할 수 있다(예를 들어, 프로세서(140)에 의한 부팅 또는 프로세싱 동안 예상되는 응답을 위해 프로세서(140)를 모니터링하여). 도 1에 관하여, 실시예에 따르면, 칩셋(125)은 연산 시스템(102)에 포함되어 컨트롤러(190)로 하여금 펌웨어(110) 및 프로세서(140) 사이의 통신 이를 테면, 디폴트 어드레스로의 접속 및 칩셋(125)을 통과하는 데이터 반환과 같은 것을 모니터링하도록 한다. 부팅하는 동안, 오류가 발생하면, 프로세서는 오류 메세지(예를 들어, "FF")를 반환 데이터로 펌웨어(110)에 보낸다. 마찬가지로, 프로세서(140)로부터의 오류 또는 다른 메시지(예를 들어, 예상되는 응답값과 비교되는 프로세서로부터의 응답)는 코어가 손상되거나 펌웨어(110) 또는 펌웨어의 영역이 손상되었다는 점, 또는 펌웨어 또는 프로세서가 단절되었다는 점을 컨트롤러(190)에 표시한다. 구체적으로, 컨트롤러(190)는 성공적인 부팅을 표시하는 프로세서에 의해 반환된 유효 데이터를 감지하기 위해, 또는 부팅 고장, "코어 손상", 펌웨어 손상, 또는 프로세서 단절을 표시하는 오류 메시지를 탐지하기 위해 부팅하는 동안 또는 부팅한 후에 칩셋(125)을 모니터링한다. 게다가, 컨트롤러(190)는 클라이언트 시스템 프로그레스를 모니터링하기 위해 연산 시스템(102)의 클라이언트 하드웨어(예를 들면, 칩셋), 컨트롤러(190)의 펌웨어 또는 이 두가지 모두를 사용할 수 있다.

<36> 블록(215)에서 펌웨어가 손상되거나 단절되었는지를 판단한다. 예를 들어, 블록(215)에서 컨트롤러(190)는 영역(120)이 이를테면 펌웨어의 피쳐 또는 프로세스에 "버그" 또는 오류를 가지게 되거나 삭제됨으로써, 손상되었는지를 판단한다. 결정 블록(215)은 블록(210)에서의 모니터링 결과(monitoring description)에 따라 판단하는

과정을 포함할 수 있다. 예를 들어, 영역(120) 또는 이 안의 구성 데이터에 따라, 스타트 업, 초기설정, 프로세서(140), 또는 오퍼레이팅 시스템의 프로그레스를 이들의 예상 응답과 비교하기 위해 블록(215)은 클라이언트 칩셋(예를 들어, 도 1의 칩셋(125))을 모니터링하는 컨트롤러(190)에 해당할 수 있다. 만약 예상 응답이 칩셋에서 탐지된 프로그레스와 일치하지 않으면, 영역(120)은 손상된 것으로 간주된다.

<37> 또한, 블록(215)은 연산 시스템(102)에 대한 스타트 업 또는 초기화 프로세스가 단절되었는지를 모니터링하는 것을 포함할 수 있다. 구체적으로, 컨트롤러(190)가 연산 시스템(102)의 하드웨어를 경유하여 영역(120)이 단절되었다고 판단하면(예를 들어, 영역(120)의 실행 동안, 연산 장치(102)의 부팅 또는 리셋 동안), 영역(120)은 단절된 것으로 간주된다.

<38> 만약 블록(215)에서 컨트롤러(190)가, 영역(120)이 손상되거나 단절되었다고 판단하면, 프로세싱은 블록(220)으로 계속된다. 블록(220)에서, 컨트롤러(190)는 클라이언트 시스템 프로세서를 중지, 중단, 또는 멈추게 하기 위해(예를 들어, 프로세서가 "중단"됨) 로직, 하드웨어 및/또는 소프트웨어 메커니즘을 사용할 수 있다. 여기서, 컨트롤러(190)는 시스템 프로세서(예를 들어, 프로세서(140), 시스템 프로세서, 또는 클라이언트 시스템의 중앙 처리 장치(CPU))로 하여금 펌웨어, 소프트웨어, 또는 오퍼레이팅 시스템 명령어의 프로세싱을 중단하도록 한다. 실시예에 따르면, 컨트롤러(190)는 또한 연산 시스템(102)의 오퍼레이팅 시스템으로 하여금 중단, 단절되게 하거나, 또는 소프트-오프 상태에 들어가게 한다.

<39> 만약 블록(215)에서 컨트롤러(190)가 영역(120)이 손상되거나 단절되지 않았다고 판단하면, 프로세싱은 블록(225)으로 계속된다. 블록(225)에서 펌웨어 및 오퍼레이팅 시스템을 실행할 수 있게 한다. 예를 들어, 블록(225)은 연산 시스템(102)의 스타트 업 또는 초기화 동안 영역(120)이 실행을 계속하거나 완성하도록하는 컨트롤러(190)에 해당할 수 있다. 블록(225) 후에 연산 장치(102)는 이를 테면, 오퍼레이팅 시스템에 컨트롤을 전함으로써, 오퍼레이팅 시스템에 따라 연산 프로세스를 실행할 수 있다. 또한, 실시예에 따라, 블록(225) 후에, 연산 장치(102)는 펌웨어(110)의 영역(130)을 실행할 수 있다(예를 들어, 오퍼레이팅 시스템에 컨트롤을 전하기에 앞서 펌웨어(110)의 실행을 완성). 예를 들어, "A"에서, 도 2에 나타난 바와 같이, 프로세스(200)는 도 3에 나타난 "A"에서 계속될 수 있다(예를 들어, 도 3에서 표시되고 설명된 바와 같이 블록(310)에서 계속됨).

<40> 블록(230)에서 컨트롤러는 이를 테면 연산 시스템(102)의 고유 식별과 같은클라이언트 시스템 특정 정보에 액세스하기 위해 로컬 데이터베이스에 액세스한다. 블록(240)에서, 컨트롤러(190)는 시스템 식별 정보를 이용하여 서버를 통한 네트워크를 통해 업데이트 데이터를 요구한다. 예를 들어, 컨트롤러(190)는 시스템 식별을 포함하여 요구하는 업데이트 펌웨어 메시지 또는 "코어 손상" 메시지를 서버(180)에 보낸다. 구체적으로, 컨트롤러(190)는 클라이언트 시스템 식별 및 문제의 식별(예를 들어, 영역(120)이 손상되었다는 표시 및/또는 업데이트된 데이터 또는 펌웨어의 요구)을 포함하는 사전정의된 메시지를 네트워크(170)를 경유하여 서버(180)에(예를 들어, 네트워크 인터페이스(160), 통신 링크(162 및 182), 및 네트워크(170)을 경유하여 서버(180)에서 실행되는 콘솔 어플리케이션에) 보낼 수 있다.

<41> 블록(245)에서 업데이트 데이터가 존재하는지 판단된다. 예를 들어, 서버(180)는 컨트롤러(190)로부터 업데이트된 펌웨어 데이터 또는 "코어 손상" 메시지 요구를 수신하고, 컨트롤러(190)에 의한 임의의 메시지가 보내진 시스템 식별에 의해 식별된 시스템에 대한 펌웨어 이미지가 존재하는지 판단하기 위해 서버(180) 내의 데이터베이스를 색인할 수 있다. 더욱 특별하게, 블록(245)에서, 서버(180)는 펌웨어 이미지에 대한 로컬 데이터베이스를 찾거나, 식별된 클라이언트 시스템, 예를 들면 연산 시스템(102)에 대한 펌웨어 이미지를 얻기 위해 IT 관리자에게 메시지를 보낸다. 만약 서버(180)가 특정 시간 안에 식별된 클라이언트에 대한 대응하는 펌웨어 이미지를 찾는다면, 서버는 "성공(successful)" 상태 메시지 및 펌웨어 이미지를 이를 테면 보안 통신, 암호화된 데이터, 및/또는 네트워크(170)에 대한 네트워크 패킷을 이용하여 연산 시스템(102)에 보낼 것이다. 반면에 만약 서버(180)가 식별된 클라이언트 시스템에 대한 적절한 펌웨어 이미지를 찾을 수 없다면, 서버(180)는 "실패(failure)" 상태 메시지를 상기 언급된 "성공" 상태 메시지에 대한 경우와 유사한 메커니즘을 사용하여 연산 시스템(102)에 보낼 것이다. 일 실시예에서 서버(180)는 식별된 클라이언트 시스템에 대한 적절한 펌웨어 이미지를 찾는데 대한 선택된 조사 시간의 타임-아웃이 될 수 있다(예를 들어, 만약 서버(180)가 데이터베이스 안에 클라이언트 시스템 식별 또는 펌웨어를 가지고 있지 않다면). 다른 실시예에서, 서버(180)는 식별된 클라이언트 시스템에 대한 적합한 펌웨어 이미지를 제공하기 위해 IT 관리자 응답을 기다리며 타임-아웃할 수 있다.

<42> 추가로, 블록(245)에서, 컨트롤러는 서버로부터 응답을 받았는지를 판단하기 위해 하드웨어를 폴링(polling)할 수 있다. 예를 들어, 컨트롤러(190)는, 서버(180)가 네트워크(170)를 통한 펌웨어 업데이트 데이터 요구에 응답을 했는지를 판단하기 위해 네트워크 인터페이스(160)와 같은 연산 시스템(102)의 하드웨어를 폴링할 수

있다. 만약, 폴링하는 동안 특정 "타임아웃" 타임 기간 내에 응답이 없으면, 프로세싱은 블록(248)으로 계속될 수 있다.

<43> 반면, 연산 시스템(102) 내에서 작동하는 컨트롤러(190)는 응답에 대하여 네트워크 인터페이스(160) 또는 서버(180)를 모니터링하고 상기 응답에 따른 적절한 조치를 취한다. 만약 블록(245)에서 업데이트 데이터가 존재하지 않거나, 또는 폴링 타임이 끝나면, 사용자가 시스템 펌웨어 오류를 통지받는 블록(248)으로 프로세싱이 계속된다. 예를 들어, 만약 컨트롤러(190)가 서버(180)로부터 고장 상태 메시지를 수신하면, 컨트롤러(190)는 연산 시스템(102)이 복구할 수 없는 시스템-펌웨어-오류 상태에 있다는 사용자 이해가능 메시지를 사용자에게 표시한다. 예를 들어, 컨트롤러는 연산 시스템(102)에 부속된 모니터로 하여금 오류 상태를 나타내는 메시지를 디스플레이하도록 할 수 있고, 연산 시스템(102)으로 하여금 오류가 존재한다고 표시하는 "경고음" 시퀀스를 출력하도록 할 수 있고, 또는 다른 방법으로 사용자에게 오류를 통지할 수 있다.

<44> 다른 대안으로서, 만약 블록(245)에서 업데이트 데이터가 존재하고 폴링 타임아웃 전에 컨트롤러에 전송된다면, 폴링 프로세싱은 블록(250)으로 계속된다. 블록(250)에서 업데이트 데이터는 네트워크를 통해 컨트롤러에 반환된다. 여기서, 블록(245)에 대해 상기 설명된 바와 같이, 서버(180)는 연산 시스템(102)에 "성공" 상태 메시지 및 펌웨어 이미지를 이를테면, 보안 통신, 암호화된 데이터, 및/또는 네트워크(170)를 통한 네트워크 패킷을 이용하여 보낸다.

<45> 블록(260)에서 업데이트 데이터는 서버로부터 네트워크를 통해 컨트롤러에 의해 수신된다. 예를 들어, 컨트롤러(190)는 "성공" 상태 메시지 및 펌웨어 업데이트 데이터를 네트워크(170)를 통해 서버(180)로부터 받을 수 있다. 펌웨어 업데이트 데이터는 영역(120)에 대한 원본의 펌웨어, 펌웨어 업데이트 데이터, 또는 펌웨어 업데이트 이미지를 포함할 수 있다.

<46> 다음으로, 블록(270)에서 펌웨어는 업데이트 데이터로 업데이트된다. 예를 들어, 컨트롤러(190)가 서버(180)로부터 성공 상태 메시지를 제공받으면, 컨트롤러(190)는 성공 상태 메시지 및 수신된 펌웨어 업데이트 데이터로 영역(120)을 프로그래밍하거나 덮어쓸 수 있다.

<47> 블록(280)에서, 펌웨어(110)는 복구되고 컨트롤러(190)는 오퍼레이팅 시스템, 연산 시스템(102), 또는 프로세서(140)에 대한 임의의 정지 또는 중단을 제거한다. 따라서, 임의의 정지 또는 중단이 제거된 후에, 시스템, 예를 들어 연산 시스템(102)은 리셋될 수 있다. 예를 들어, 블록(280)은 업데이트 되고, 손상되지 않고, 적절한 영역(120)을 가진 연산 장치(102)를 리부팅하기 위해 프로세스(200)의 블록(202)으로 돌아갈 수 있다.

<48> 더욱이, 실시예에 따르면, 영역(120)은 영역(130)이 손상되었는지 판단하기 위해 영역(130)을 모니터링할 수 있다. 영역(130)을 모니터링하는 영역(120)의 프로세스는 독립적으로 발생하거나 또는 영역(120)을 모니터링하는 컨트롤러(190)를 포함하지 않는 시스템에서 발생할 수 있다. 그러나, 상기와 같은 시스템에서, 컨트롤러(190)는 서버(180)로부터 영역(130)에 대한 펌웨어 업데이트 데이터를 요구 및 수신하기 위해 여전히 필요할 것이다. 반대로, 영역(120)이 영역(130)을 모니터링하는 프로세스는, 컨트롤러(190)가 영역(120)을 모니터링하는 상기 내용을 추가하는 실시예가 될 수 있는데, 이를 테면 컨트롤러(190)가 영역(120)을 모니터링한 후에 영역(120)이 영역(130)을 모니터링하는 것과 같다.

<49> 예를 들어, 도 3은 네트워크를 통해 펌웨어를 업데이트하는 프로세스의 흐름도(300)이다. 블록(302)에서 시스템은 부팅되거나 리셋된다. 예를 들어, 블록(302)은 블록(202)에 대한 상기 설명에 대응할 수 있다. 다음으로, 블록(303)에서 컨트롤러는 부팅, 초기화, 펌웨어 실행, 오퍼레이팅 시스템 실행, 또는 프로세싱을 계속하는 시스템의 능력과 관계없이 개시되거나 초기화된다. 예를 들어, 블록(303)은 블록(203)에 대한 상기 설명에 대응할 수 있다. 블록(305)에서 컨트롤러(190)와 같은 컨트롤러는, 연산 시스템(102)의 스타트 업과 같은 클라이언트 시스템 스타트 업으로 통지된다. 예를 들어, 블록(305)은 블록(205)에 대한 상기 설명에 대응할 수 있다.

<50> 블록(310)에서 시스템은 펌웨어가 손상되었는지를 판단하기 위해 모니터링된다. 예를 들어, 영역(120)은 영역(130)이 손상되었는지 모니터링하거나, 판단하거나, 발견하기 위해 컨트롤러(190)의 펌웨어에 내장된 방법S 또는 로직을 포함할 수 있다.

<51> 상기 언급된 바와 같이, 프로세스(300)의 블록(310)은 도 2에 나타난 바와 같이 블록(225) 다음에 수행될 수 있다. 따라서, 도 2에 나타난 바와 같이 프로세스(200)가 영역(120)을 이용하여 적절한 부팅을 제공한 후에, 블록(310)에서 시작하는 프로세스(300)는 영역(130)이 손상되었는지를 판단하기 위해 수행된다.

<52> 예를 들어, 블록(310)에서, 영역(120)은 영역(130)이 손상되었는지를 판단하기 위해 프로세서(140), 연산 시스

템(102) 상에서 운영되는 오퍼레이팅 시스템, 상기 언급된 클라이언트 칩셋, 예를 들어 도 1의 칩셋(125)을 모니터링할 수 있다. 더욱이, 블록(310)에서 영역(120)은 영역(130)이 손상되었는지 판단하기 위해 체크섬-기반 체크, 서명-기반 체크, 및/또는 하드웨어-기반 체크를 수행할 수 있다. 블록(310)은 영역(130)이 손상되었는지 판단하거나 탐지하기 위해 블록(210)에 대해 기술된 바와 같이 컨트롤러(190)에 의해 칩셋(125)을 모니터링하는 것 및/또는 연산 시스템(102)의 오퍼레이팅 시스템을 모니터링하는 것을 포함할 수 있다.

<53> 블록(315)에서 펌웨어가 손상되거나 단절되었는지를 판단한다. 예를 들어, 블록(315)에서 영역(120)은 영역(130)이 이를테면 펌웨어의 피쳐 또는 프로세스에 "버그" 또는 오류를 가지게 되거나 삭제됨으로써, 손상되었는지를 판단한다. 결정 블록(315)은 블록(310)에서의 모니터링 결과에 따라 판단하는 과정을 포함할 수 있다. 예를 들어, 영역(130) 또는 그 구성 데이터에 따라 스타트 업, 초기설정, 프로세서(140), 또는 오퍼레이팅 시스템의 프로그레스를 이들의 예상 응답과 비교하기 위해 블록(315)은 클라이언트 칩셋, 예를 들어, 도 1의 칩셋(125)을 모니터링하는 컨트롤러(190) 또는 영역(120)에 해당할 수 있다. 만약 예상 응답이 칩셋에서 탐지된 프로그레스와 일치하지 않으면(예를 들어, 오류 또는 "FF" 메시지를 포함함), 영역(130)은 손상된 것으로 간주된다.

<54> 더욱이, 블록(315)은 프로세서(140)가 중단 또는 정지되는지를 판단하는 것, 연산 시스템(102)에 대한 스타트 업 또는 초기화 프로세스가 단절되는지를 판단하는 것, 연산 시스템(102)에서 작동하는 오퍼레이팅 시스템이 정지되거나, 단절되거나, 또는 소프트-오프되는지를 판단하는 것, 및/또는 연산 시스템(102)의 펌웨어, 예를 들어 펌웨어(110)가 손상되거나 단절되는지를 판단하는 것을 포함한다. 예를 들어, 블록(315)은 영역(130)의 체크섬-기반 체크, 서명-기반 체크, 및/또는 하드웨어-기반 체크를 수행하거나 모니터링하는 영역(120)에 해당할 수 있다.

<55> 만약 블록(315)에서 영역(130)이 손상되지 않았다고 판단되면, 프로세싱은 블록(325)으로 계속된다. 블록(325)에서 펌웨어 및 오퍼레이팅 시스템은 실행할 수 있게 된다. 예를 들어, 블록(325)은 연산 시스템(102)의 스타트 업 또는 초기화 동안 영역(130)이 실행을 계속하거나 완성하도록 하는 영역(120) 및 컨트롤러(190)에 해당할 수 있다. 따라서, 블록(325)은 이를 테면 오퍼레이팅 시스템에 컨트롤을 전함으로써, 오퍼레이팅 시스템에 따라 연산 프로세스를 실행하도록 된 연산 장치(102)에 해당할 수 있다. 또한, 실시예에 따라, 블록(325) 후에, 연산 장치(102)는 펌웨어(110), 영역(120), 및/또는 영역(130)을 업데이트하기 위해 특정 시간에 또는 주기적으로 예정된 업데이트와 같은 자동 업데이트를 체크할 수 있다. 예를 들어, 도 3에 도시된 "B"에서, 프로세스(300)는 도 4에 도시된 "B"로 계속될 수 있다(예를 들어, 도 4에 관하여 아래에 표시되고 기술된 바와 같이 블록(415)에서 계속됨).

<56> 다른 대안으로서, 블록(315)에서 영역(130)이 손상된 것으로 판단되면, 프로세싱은 블록(318)으로 계속된다. 블록(318)에서 컨트롤러와 통신하는 하드웨어는 초기화된다. 예를 들어, 영역(130)이 손상되었다고 영역(120)이 판단하면, 영역(120)은 컨트롤러(190)와 통신하고 배열하기 위해 필요한 최소한의 하드웨어 세트를 초기화하기 위해 하드-코딩된 시스템 파라미터를 사용할 수 있다. 영역(120)은 그리고 나서 영역(130)이 손상되었다고 식별하는 파라미터와 함께 사전정의된 메시지를 컨트롤러(190)에 보낸다. 마찬가지로, 영역(120)은 영역(120 및/또는 130)의 업데이트 또는 회복을 요청하는 파라미터와 함께 사전정의된 메시지를 컨트롤러(190)에 보낸다. 영역(120)은 그리고 나서 상기 기술된 하드-코딩된 시스템 파라미터로 초기화된 하드웨어 상의 컨트롤러(190)로부터의 응답을 폴링한다.

<57> 블록(318) 후에, 프로세스(300)는 클라이언트 시스템 식별이 액세스되는 블록(330)으로 계속된다. 블록(330)은 클라이언트 시스템(예를 들어, 연산 시스템(102))에 대한 식별을 액세스하는 컨트롤러(190)에 해당할 수 있다. 예를 들어, 블록(330)은 상기 기술된 블록(230)에 해당할 수 있다.

<58> 블록(330) 후에, 프로세싱은 네트워크를 통해 업데이트 펌웨어 데이터가 요구되는 블록(340)으로 계속된다. 예를 들어, 컨트롤러(190)는 블록(340)에 대해 상기 기재된 메시지 및 시스템 식별 정보를 이용하여, 네트워크(170)를 통해 서버(180)로부터 영역(120 및/또는 130)에 대한 업데이트 데이터를 요구할 수 있다. 또한, 블록(340)에 보내진 시스템 식별 정보는, 요구되는 펌웨어가 펌웨어(110)의 영역(120 및/또는 130)을 위한 것이라는 식별을 포함할 수 있다.

<59> 블록(340) 후에, 프로세싱은 업데이트 데이터가 존재하는지가 판단되는 블록(345)으로 계속된다. 블록(345)은 상기 기술된 블록(245)에 대응한다. 예를 들어, 서버(180)는 제공된 시스템 식별에 대응하는 적절한 업데이트 펌웨어 데이터를 찾지 못할 수 있거나, 또는 컨트롤러(190)가 하드웨어를 폴링하는데 타임 아웃될 수 있다.

<60> 반면, 연산 시스템(102)에서 작동하는 컨트롤러(190)는 응답을 위해 네트워크 인터페이스(160) 또는 서버(180)를 모니터링하고 상기 응답에 따라 적절한 조치를 취한다. 만약 결정 블록(345)에서, 업데이트가 존재하지 않으면, 프로세싱은 블록(248)에 대하여 상기 기술된 것과 같이 사용자가 시스템 펌웨어 오류를 통지받는 블록(348)으로 계속된다. 추가로, 블록(348)에서, 컨트롤러(190)는 영역(130)에 대한 펌웨어를 얻지 못했다는 것을 나타내는 오류 응답을 영역(120)에 보낸다. 결과적으로, 영역(120)은 컨트롤러(190)로부터 고장 상태 메시지를 수신하고, 이를 테면 디스플레이, 경고음, 또는 연산 시스템(102)에 대해 가능한 다른 방법 등을 통해 사용자에게 통지할 수 있다.

<61> 만약 블록(345)에 업데이트 데이터가 존재하면, 프로세싱은 업데이트 데이터가 네트워크를 통해 컨트롤러에 반환되는 블록(350)으로 계속된다. 블록(350)은 상기 기술된 블록(250)에 해당할 수 있다. 프로세싱은 업데이트 데이터가 서버로부터 네트워크를 통해 컨트롤러에 의해 수신되는 블록(360)으로 계속된다.

<62> 블록(360)에서 업데이트 데이터는 서버로부터 네트워크를 통해 컨트롤러에 의해 수신된다. 예를 들어, 컨트롤러(190)는 네트워크(170)를 통해 서버(180)로부터 "성공" 상태 메시지 및 펌웨어 업데이트 데이터를 수신할 수 있다. 펌웨어 업데이트 데이터는 영역(120 및/또는 130)에 대한 원본의 펌웨어, 펌웨어 업데이트 데이터, 또는 펌웨어 업데이트 이미지를 포함할 수 있다. 구체적으로, 컨트롤러(190)는 서버(180)로부터 영역(120 및/또는 130)에 대한 업데이트 데이터를 다운로드 받고, 업데이트 데이터를 시스템 메모리(예를 들어, 메모리(142))에 복사하고, 영역(120)에 성공 상태 메시지를 보낼 수 있다. 다른 대안으로서, 컨트롤러(190)는 영역(120 및/또는 130)에 직접 업데이트 데이터를 다운로드 받고(예를 들어, 영역(120 및/또는 130)의 현재 있는 데이터를 업데이트 데이터로 덮어씌움) 영역(120)에 성공 상태 메시지를 보내기 위해, 콘솔 또는 서버 통과 인터페이스를 마련할 수 있다.

<63> 다음으로, 블록(370)에서 펌웨어는 업데이트 데이터로 업데이트된다. 예를 들어, 컨트롤러(190)가 서버(180)로부터 성공 상태 메시지를 받으면, 컨트롤러(190)는 성공 상태 메시지와 펌웨어 업데이트 데이터로 영역(120 및/또는 130)을 프로그래밍하거나 덮어쓸 수 있다. 더욱이, 만약 영역(120)이 성공 상태 메시지를 수신하면, 영역(120)은 메모리(예를 들어, 메모리(142))에 저장된 업데이트 데이터를 읽을 수 있거나, 또는 영역(120 및/또는 130)에 업데이트 데이터를 프로그램하거나, 쓰거나, 또는 덮어쓰기 위해 상기 기술된 통과 인터페이스로부터 데이터를 읽거나 참작할 수 있다.

<64> 블록(380)에서, 펌웨어(110)는 복구되고 컨트롤러(190) 및/또는 영역(120)은 클라이언트 시스템을 리셋한다. 예를 들어, 영역(120)은 시스템 특정 프로세스를 이용하여 연산 시스템(102)을 리셋할 수 있다. 예를 들어, 블록(380)은 업데이트 되고, 손상되지 않고, 적절한 영역(130)을 가진 연산 장치(102)를 리부팅하기 위해 프로세스(300)의 블록(302)으로 돌아갈 수 있다.

<65> 따라서, 블록(230, 240, 245, 248, 250, 260, 및 270)은 영역(130)을 덮어쓰거나 업데이트 하기 위해 상기 블록에서 취해지는 조치를 영역(130)이 조절함으로써, 영역(130)만을 액세스하고, 요구하고, 조사하고, 반환하고, 수신하고, 업데이트하는데 적용될 수 있다. 다른 대안으로서, 상기 블록은 영역(120) 및 영역(130)에 적용될 수 있는데, 이를 테면 동일한 세트의 조치 동안에, 영역(120 및 130) 모두를 덮어쓰거나 업데이트하기 위해 컨트롤러(190)가 상기 동작을 컨트롤함으로써 이루어진다.

<66> 추가로, 실시예에 따르면, 펌웨어(110)는 서버(180)로부터 업데이트를 체크하고 수신함으로써 특정 시간에 또는 주기적으로 업데이트되는 것과 같이 자동으로 업데이트될 수 있다(예를 들어, 영역(120) 및/또는 영역(130)을 업데이트). 펌웨어를 자동으로 업데이트하는 프로세스는 독립적으로 일어나거나, 영역(120)을 모니터링하는 컨트롤러(190) 및/또는 영역(130)을 모니터링하는 영역(120)을 포함하지 않는 시스템에서 일어날 수 있다. 그러나, 상기와 같은 시스템에서, 컨트롤러(190)는 서버(180)로부터 펌웨어(110)에 대한 펌웨어 업데이트 데이터를 요구 및 수신하기 위해 여전히 필요할 것이다. 반대로, 펌웨어(110)의 자동 업데이트를 가지는 프로세스는 영역(120)을 모니터링하는 컨트롤러(190) 및/또는 영역(120)이 영역(130)을 모니터링하는 상기 설명을 포함하는 실시예가 될 수 있다.

<67> 예를 들어, 도 4는 네트워크를 통해 펌웨어를 자동으로 업데이트하는 프로세스의 흐름도(400)이다. 블록(402)에서 시스템은 부팅되거나 리셋된다. 예를 들어, 블록(402)은 블록(202)에 대한 상기 설명에 해당할 수 있다. 다음으로, 블록(403)에서 컨트롤러는 부팅, 초기화, 펌웨어 실행, 오퍼레이팅 시스템 실행, 또는 프로세싱을 계속하는 시스템의 능력과 관계없이 개시되거나 초기화된다. 예를 들어, 블록(403)은 블록(203)에 대한 상기 설명에 해당할 수 있다. 블록(405)에서 컨트롤러(190)와 같은 컨트롤러는, 연산 시스템(102)의 스타트 업과 같은 클라이언트 시스템 스타트 업으로 통지된다. 예를 들어, 블록(405)은 블록(205)에 대한 상기 설명에 해당할 수

있다.

<68> 블록(415)에서 시스템은 펌웨어(110)의 전부 또는 일부의 자동 업데이트를 할 시간인지를 판단하기 위해 모니터링된다. 실시예에 따르면, 컨트롤러(190) 또는 영역(120)은 선택된 시간인지, 또는 마지막 자동 업데이트 또는 시도 이후로 선택된 기간이 만료되었는지를 판단하기 위해, 클럭 신호를 모니터링하는 로직을 포함할 수 있다. 따라서, 블록(415)은 부팅, 프로세서(190), 또는 연산 시스템(102)의 오퍼레이팅 시스템이 실행, 정지, 중단, 단절, 또는 소프트-오프 되는지에 상관없이 일어날 수 있다. 선택된 시간 또는 기간이 만료되면, 영역(120) 및/또는 영역(130)의 자동 업데이트 시간이 된다.

<69> 상기 언급된 바와 같이, 프로세스(400)의 블록(415)은 도 3에 나타난 블록(325) 후에 일어날 수 있다. 따라서, 도 3에 나타난 바와 같이 프로세스(300)가 영역(130)을 이용하여 적절한 부팅을 제공한 후에, 블록(415)에서 시작하는 프로세스(400)는 자동 업데이트가 필요한지를 판단하기 위해 수행될 수 있다.

<70> 만약 블록(415)에서 펌웨어의 자동 업데이트 시간이 아니면, 프로세싱은 블록(425)으로 계속된다. 블록(425)에서 펌웨어 및 오퍼레이팅 시스템이 실행될 수 있다. 예를 들어, 블록(425)은 연산 시스템(102)의 스타트업 또는 초기화가 계속되도록 하는 컨트롤러(190) 및 펌웨어(110)에 해당할 수 있다. 블록(425)은 연산 장치(102)의 오퍼레이팅 시스템이 작동하는 때에 해당할 수 있고, 따라서 연산 장치(102)가 오퍼레이팅 시스템에 따라 연산 프로세스 실행을 계속하도록 할 수 있다.

<71> 다른 대안으로서, 만약 블록(415)에서 펌웨어의 자동 업데이트 시간이 되면, 프로세싱은 블록(418)으로 계속된다. 블록(418)에서 컨트롤러와 통신하는 하드웨어는 초기화된다. 예를 들어, 자동 업데이트 시간이 되면, 영역(120)은 컨트롤러(190)와 통신하고 컨트롤러를 구성하는데 필요한 최소한의 하드웨어 세트를 초기화하기 위해 하드-코딩된 시스템 파라미터를 사용할 수 있다. 영역(120)은 그리고 나서 영역(120) 및/또는 영역(130)의 자동 업데이트 시간이라고 식별하는 파라미터와 함께 사전정의된 메시지를 컨트롤러(190)에 보낸다.

<72> 다른 대안으로서, 영역(120)은 영역(130)을 업데이트하거나 복구하기 위해 컨트롤러(190)에 영역(103)이 손상되었다는 메시지 및/또는 요구를 보낼 수 있다. 이 경우에서, 블록(418)은 상기의 블록(318)에 해당할 수 있다. 영역(120)은 그리고 나서 상기 기술된 하드-코딩된 시스템 파라미터에 의해 초기화된 하드웨어 상의 컨트롤러(190)로부터의 응답을 폴링한다. 예를 들어, 영역(120)은 서버(180)가 응답하는지를 판단하기 위해 연산 시스템(102)의 하드웨어를 폴링함으로써(예를 들면, 네트워크 인터페이스(160)에 의해) 네트워크 인터페이스(160) 또는 컨트롤러(190)를 모니터링할 수 있다.

<73> 블록(418) 후에, 프로세스(400)는 클라이언트 시스템 식별을 액세스하는 블록(430)으로 계속된다. 블록(430)은 클라이언트 시스템(예를 들어, 연산 시스템(102)) 및/또는 펌웨어가 자동으로 업데이트 되기 위한 식별에 액세스하는 컨트롤러(190)에 해당할 수 있다. 예를 들어, 블록(430)은 상기 기술된 블록(230) 또는 블록(330)에 해당할 수 있다. 더욱이, 블록(430)에서, 자동 업데이트 시간이 되면, 컨트롤러(190)는 클라이언트 시스템 식별 및 펌웨어(110)에 대한 수정 정보, 배포 데이터 등을 검색하기 위해 자신의 로컬 데이터베이스에 접속할 수 있다.

<74> 블록(440)에서, 컨트롤러(190)는 시스템 식별 정보를 이용하여 네트워크를 통해 서버에 업데이트 데이터를 요구한다. 예를 들어, 컨트롤러(190)는 시스템 식별을 포함하여 자동 업데이트 메시지, 요구 펌웨어 업데이트 메시지, 또는 "코어 손상" 메시지를 서버(180)에 보낼 수 있다.

<75> 구체적으로, 컨트롤러(190)는 클라이언트 식별 정보 뿐만 아니라 수정 정보, 배포 데이터 등(예를 들어, 영역(120) 및/또는 영역(130)에 대한 수정 정보, 배포 데이터 등)을 포함하는 사전정의된 메시지를 서버(180)에 보낸다. 컨트롤러(190)로부터 서버(180)로의 메시지의 통신은 블록(240)의 상기 기술에 대응할 수 있다.

<76> 결과적으로, 블록(445)에서, 서버(180)는 컨트롤러(190)로부터 받은 메시지를 자동 업데이트 요구로 해석하고, 컨트롤러(190)로부터의 메시지에 있는 수정 정보, 배포 데이터 등에 대한 최신 또는 차후의 수정을 위해 서버(180)의 데이터베이스를 검색한다. 다른 대안으로서, 블록(445)에서, 서버(180)는 클라이언트 시스템에 대한 이미지를 요구하는 메시지를 IT 관리자에게 보낼 수 있다.

<77> 반면, 연산 시스템(102)에서 작동하는 컨트롤러(190)는 응답을 위해 네트워크 인터페이스(160) 또는 서버(180)를 모니터링하고, 상기 응답에 따라 적절한 조치를 취한다. 데이터베이스를 색인하고, 폴링하고, 서버(180)에 업데이트 데이터가 존재하는지를 판단하는 방법은 블록(345)에 대한 상기 기술에 대응할 수 있다. 예를 들어, 만약 결정 블록(445)에서, 영역(120)에 의한 폴링이 컨트롤러(190)가 선택된 타임아웃 기간 전에 서버(180)로부

터 메시지를 수신하지 않았다고 판단하면, 프로세싱은 블록(448)으로 계속되는데 여기서 펌웨어(110)는 이를 테면 연산 시스템(102)으로부터의 디스플레이된 메시지를 통해 "네트워크 입수 불가능"(예를 들어, 네트워크(170))을 나타내는 메시지를 사용자에게 보낸다.

<78> 또한, 만약 결정 블록(445)에서, 이를 테면 서버가 선택된 타임아웃 기간 전에 서버에서 매치를 찾지 못하는 등, 업데이트가 존재하지 않으면, 서버(180)는 컨트롤러(190)에 "입수가능한 자동 업데이트 없음"을 나타내는 메시지를 보낸다. 그리고 나서, 프로세스(400)는 프로세서(190)가 사용자에게, 이를 테면 연산 시스템(102)으로부터의 디스플레이된 메시지를 통해, "입수가능한 자동 업데이트 없음"을 통지하는 블록(448)으로 계속된다.

<79> 게다가, 블록(448)에서, 컨트롤러(190)는 펌웨어(110)에 "입수가능한 자동 업데이트 없음" 메시지를 보낼 수 있다. 펌웨어(110)가 "입수가능한 자동 업데이트 없음" 메시지를 수신한 후에, 펌웨어(110)는 사용자에게 연산 시스템(102)으로부터의 디스플레이된 메시지를 통해 "입수가능한 자동 업데이트 없음" 메시지를 통지할 수 있다. 블록(448)에서, 펌웨어(110)가 "입수가능한 자동 업데이트 없음", 또는 "네트워크 입수 불가능" 메시지를 수신한 후에는, 프로세스(400)가 자동 업데이트(예를 들어, 펌웨어가 손상되거나 단절되지 않았다고 가정)에 대한 것이기 때문에, 펌웨어(110)는 연산 시스템(102)으로 하여금 부팅, 스타트업, 초기화, 또는 오퍼레이팅 시스템을 실행하는 것을 계속하게 할 수 있다.

<80> 만약 블록(445)에서 업데이트 데이터가 존재하면, 프로세싱은 업데이트 데이터가 네트워크를 통해 컨트롤러에 반환되는 블록(450)으로 계속된다. 블록(450)은 블록(250)의 상기 기술에 대응할 수 있다. 예를 들어, 블록(450)에서, 서버(180)는 "성공" 상태 메시지 및 적합한 최근 수정을 연산 시스템(102)에 대한 펌웨어 이미지에 반환할 수 있는데, 이를테면 보안 서버 세션, 암호화된 데이터 및/또는 네트워크 패킷을 이용하여 이루어진다. 프로세싱은 그리고 나서 업데이트 데이터가 서버로부터 네트워크를 통해 컨트롤러에 의해 수신되는 블록(460)으로 계속된다.

<81> 블록(460)에서 업데이트 데이터는 서버로부터 네트워크를 통해 컨트롤러에 의해 수신된다. 예를 들어, 컨트롤러(190)는 네트워크(170)를 통해 서버(180)로부터 "성공" 상태 메시지 및 펌웨어 업데이트 데이터를 수신할 수 있다. 펌웨어 업데이트 데이터는 펌웨어(110)의 전부 또는 일부 영역에 대한 원본의 펌웨어, 펌웨어 업데이트 데이터, 또는 펌웨어 업데이트 이미지를 포함할 수 있다. 구체적으로, 컨트롤러(190)는 서버(180)로부터 펌웨어(110)에 대한 업데이트 데이터를 다운로드 받고, 업데이트 데이터를 시스템 메모리(예를 들어, 메모리(142))에 복사하고, 영역(120)에 성공 상태 메시지를 보낼 수 있다. 다른 대안으로서, 컨트롤러(190)는 펌웨어(110)에 직접 업데이트 데이터를 다운로드 받고(예를 들어, 펌웨어(110)의 현재 있는 데이터를 업데이트 데이터로 덮어씌움) 영역(120)에 성공 상태 메시지를 보내기 위해, 콘솔 또는 서버 통과 인터페이스를 마련할 수 있다.

<82> 다음으로, 블록(470)에서 펌웨어는 업데이트 데이터로 업데이트된다. 예를 들어, 컨트롤러(190)가 서버(180)로부터 성공 상태 메시지를 받으면, 컨트롤러(190)는 성공 상태 메시지와 펌웨어 업데이트 데이터로 펌웨어(110)(예를 들어, 영역(120) 및 영역(130))를 프로그래밍하거나 덮어쓸 수 있다. 더욱이, 만약 영역(120)이 성공 상태 메시지를 수신하면, 영역(120)은 메모리(예를 들어, 메모리(142))에 저장된 업데이트 데이터를 읽을 수 있거나, 또는 업데이트 데이터로 펌웨어(110)를 프로그래밍하거나, 쓰거나, 또는 덮어쓰기 위해 상기 기술된 통과 인터페이스로부터 업데이트 데이터를 읽거나 참작할 수 있다. 여기서, 응답을 위해 하드웨어를 폴링하는 영역(120)은, 시스템 메모리, 예를 들어 메모리(142)로부터나 서버 통과 인터페이스를 통해서, 성공 상태 메시지 및 펌웨어 이미지의 새로운 또는 최근 버전을 수신하고, 업데이트가 영역(130)에 쓰여지도록 할 것이다.

<83> 블록(480)에서, 펌웨어(110)는 업데이트되고, 컨트롤러(190) 및/또는 영역 펌웨어(110)는 클라이언트 시스템을 리셋한다. 예를 들어, 영역(120)은 시스템 특정 프로세스를 이용하여 연산 시스템(102)을 리셋할 수 있다. 블록(480)은 자동으로 업데이트된 펌웨어(110)로 연산 장치(102)를 리부팅하기 위해 프로세스(400)의 블록(402)으로 돌아가는 것을 고려할 수 있다.

<84> 상기 명세서에는, 특정 실시예가 기술되어 있다. 그러나, 청구항에서 설명된 실시예의 사상 및 범위를 벗어나지 않는 범위 내에서 다양한 수정 및 변경을 여기에 가할 수 있다. 따라서, 명세서 및 도면은 제한된 것으로 보기보다는 예시적인 것으로 간주되어야 한다.

## *도면의 간단한 설명*

<4> 도 1은 네트워크를 통해 펌웨어를 셀프-모니터링하고 업데이트하는 시스템의 블록도.

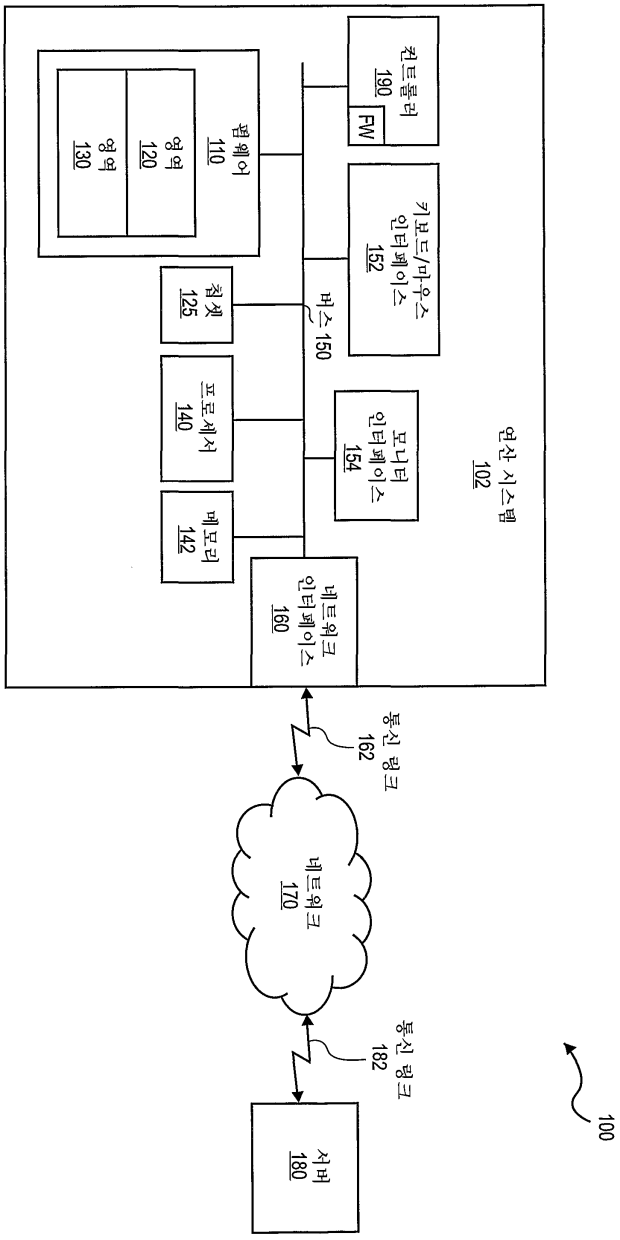
<5> 도 2는 네트워크를 통해 펌웨어를 업데이트하는 프로세스의 흐름도.

<6> 도 3은 네트워크를 통해 펌웨어를 업데이트하는 프로세스의 흐름도.

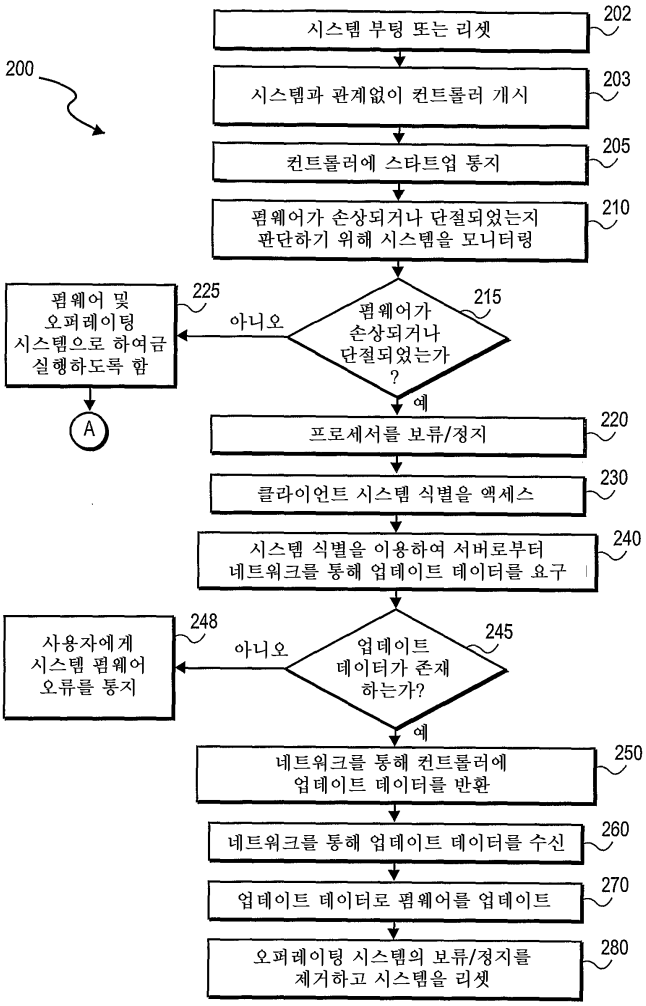
<7> 도 4는 네트워크를 통해 펌웨어를 자동으로 업데이트하는 프로세스의 흐름도.

## 도면

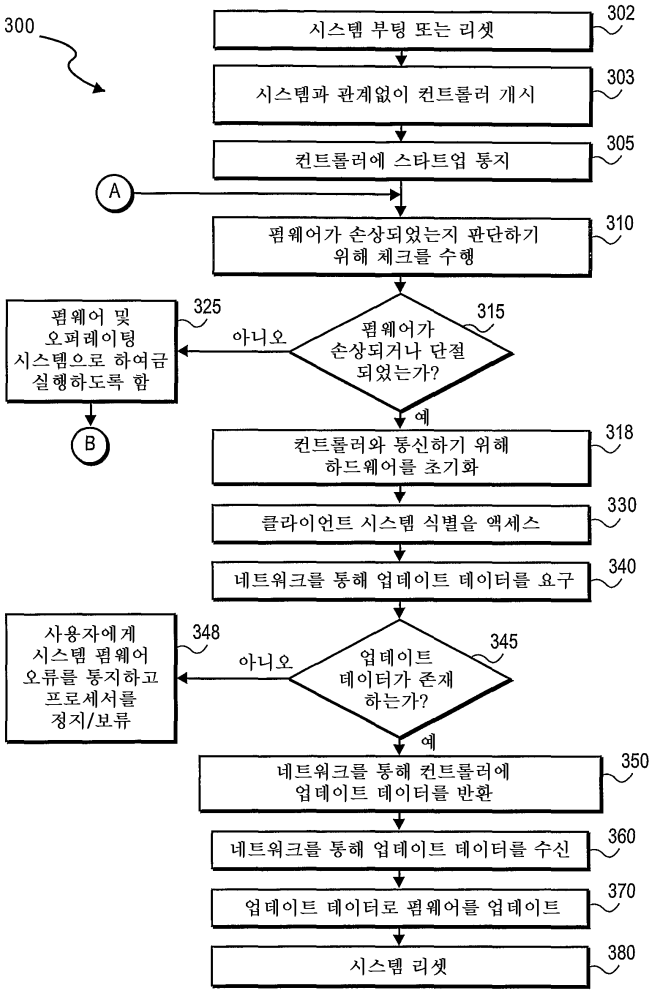
### 도면1



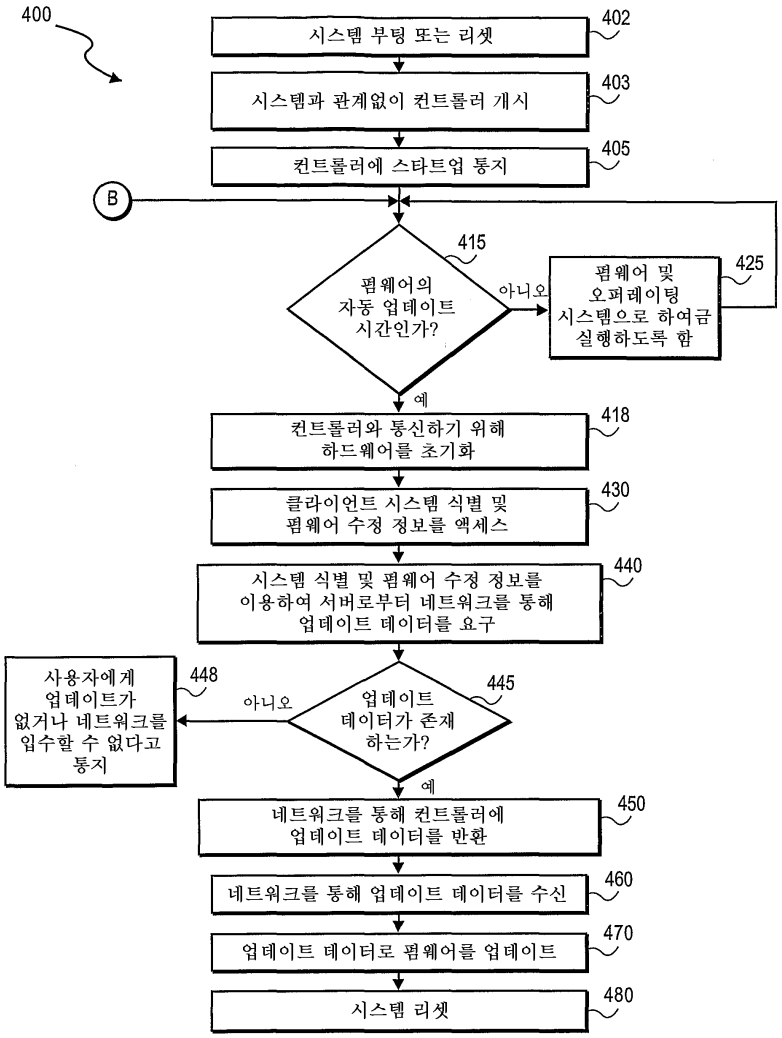
도면2


200
시스템 부팅 또는 리셋
202
시스템과 관계없이 컨트롤러 개시
203
컨트롤러에 스타트업 통지
205
펌웨어가 손상되거나 단절되었는지 판단하기 위해 시스템을 모니터링
210
펌웨어가 손상되거나 단절되었는가 ?
215
아니오
펌웨어 및 오퍼레이팅 시스템으로 하여금 실행하도록 함
225
A
예
프로세서를 보류/정지
220
클라이언트 시스템 식별을 액세스
230
시스템 식별을 이용하여 서버로부터 네트워크를 통해 업데이트 데이터를 요구
240
업데이트 데이터가 존재하는가?
245
아니오
사용자에게 시스템 펌웨어 오류를 통지
248
예
네트워크를 통해 컨트롤러에 업데이트 데이터를 반환
250
네트워크를 통해 업데이트 데이터를 수신
260
업데이트 데이터로 펌웨어를 업데이트
270
오퍼레이팅 시스템의 보류/정지를 제거하고 시스템을 리셋
280

도면3


300
시스템 부팅 또는 리셋
302
시스템과 관계없이 컨트롤러 개시
303
컨트롤러에 스타트업 통지
305
A
펌웨어가 손상되었는지 판단하기 위해 체크를 수행
310
펌웨어가 손상되거나 단절되었는가?
315
아니오
펌웨어 및 오퍼레이팅 시스템으로 하여금 실행하도록 함
325
B
예
컨트롤러와 통신하기 위해 하드웨어를 초기화
318
클라이언트 시스템 식별을 액세스
330
네트워크를 통해 업데이트 데이터를 요구
340
업데이트 데이터가 존재하는가?
345
아니오
사용자에게 시스템 펌웨어 오류를 통지하고 프로세서를 정지/보류
348
예
네트워크를 통해 컨트롤러에 업데이트 데이터를 반환
350
네트워크를 통해 업데이트 데이터를 수신
360
업데이트 데이터로 펌웨어를 업데이트
370
시스템 리셋
380

도면4


400
시스템 부팅 또는 리셋
402
시스템과 관계없이 컨트롤러 개시
403
컨트롤러에 스타트업 통지
405
B
415
펌웨어의 자동 업데이트 시간인가?
아니오
펌웨어 및 오퍼레이팅 시스템으로 하여금 실행하도록 함
425
예
컨트롤러와 통신하기 위해 하드웨어를 초기화
418
클라이언트 시스템 식별 및 펌웨어 수정 정보를 액세스
430
시스템 식별 및 펌웨어 수정 정보를 이용하여 서버로부터 네트워크를 통해 업데이트 데이터를 요구
440
업데이트 데이터가 존재하는가?
445
아니오
사용자에게 업데이트가 없거나 네트워크를 입수할 수 없다고 통지
448
예
네트워크를 통해 컨트롤러에 업데이트 데이터를 반환
450
네트워크를 통해 업데이트 데이터를 수신
460
업데이트 데이터로 펌웨어를 업데이트
470
시스템 리셋
480